P.Texto(01).dot

MÓDULO

INTRODUCCIÓN A LOS AUTÓMATAS PROGRAMABLES S7-1200

UNIDAD 3

ÁREAS DE MEMORIA

UD000082_V(01)

Instituto Tecnológico

ÍNDICE

Motivación 3

Propósitos 4

Preparación para la unidad 5

1. Introducción 7

2. La información en el autómata 9

2.1. La unidad básica de información. El bit 9

2.1.1. ¿Qué es un bit? 9

2.1.2. El bit y los sistemas electrónicos 11

2.1.3. La información estado-acción 12

2.2. Agrupaciones de bits 15

2.2.1. El *byte* 15

2.2.2. *Word* 18

2.2.3. Doble *word* 19

3. El interior del autómata 20

3.1. Bloques componentes del autómata 20

3.2. Dispositivos de entrada/salida 21

3.3. La CPU 21

3.3.1. La CPU y la memoria 22

3.3.2. Funcionamiento de la CPU 22

3.3.2.1. Recursos de la CPU 23

3.3.2.2. El ciclo de la CPU 24

3.4. La memoria 25

3.4.1. El *byte* como unidad fundamental 25

3.4.2. Almacenamiento del programa en la memoria 28

3.4.3. Zonas de memoria del autómata 30
3.4.3.1. Imagen del proceso de entradas-salidas 31
3.4.4. Sistema operativo y datos del autómata 32
3.4.4.1. Memoria de programa 32
3.4.5. Tipos de memoria 33
3.4.5.1. Memoria ROM (*read only memory*, memoria de solo lectura) 33
3.4.5.1.1. PROM (ROM programable) 33
3.4.5.1.2. EPROM (*erasable programmable ROM*, ROM borrable y programable) 33
3.4.5.1.3. EEPROM (*electrically erasable programmable ROM*, ROM programable borrable eléctricamente) 34
3.4.5.1.4. Memoria *flash* EPROM 34
3.4.5.2. Memoria RAM (*random acces memory*, memoria de acceso aleatorio) 35
3.4.5.2.1. SRAM (*static RAM* o RAM estática) 35
3.4.5.2.2. DRAM (*dynamic RAM* o RAM dinámica) 35
3.4.5.2.3. NVRAM (*non volatile RAM*, RAM no volátil) 35
3.5. El bus 36
3.5.1. El bus de datos 36
3.5.2. El bus de direcciones 37
3.6. Funcionamiento autómata 37
Conclusiones 39
Recapitulación 40
Autocomprobación 41
Solucionario 45
Propuestas de ampliación 46
Bibliografía 47

MOTIVACIÓN

Vamos a adentrarnos en el conocimiento del autómata. El autómata es una máquina compleja y su estudio tiene muchos “frentes”.

En esta unidad vamos a conocer la forma en que maneja la información nuestro autómata. El estudio de esta parte del autómata programable resulta vital para conocer cómo funciona el mismo. Básicamente, el autómata trabaja con información, recibe información del entorno y es programado para tomar decisiones en función de esa información.

Esta información debe ser entendida por el autómata y debe ser procesada por él mismo.

PROPÓSITOS

Con esta unidad conocerás:

- La relación del autómata programable y el entorno que le rodea en una automatización.

  El autómata interacciona con su entorno de automatización llevando a cabo accionamientos en base a un programa y a una información que dicho entorno le proporciona.

- Cómo se codifica la información para que el autómata la entienda. Cómo se expresa la información en un entorno digital.

- Cómo el autómata obtiene la información, cómo la procesa, cómo la almacena y cómo expresa los resultados del programa en el exterior.

PREPARACIÓN PARA LA UNIDAD

Seguimos trabajando con el bit y el sistema de numeración binario. Hay que remarcar que la información resulta ser un eje central sobre el que gira el funcionamiento del autómata. La información consiste en datos y los datos para las máquinas electrónicas son una sucesión de unos y ceros.

En consecuencia, deberemos tener muy presente la numeración en sistema binario.

1. INTRODUCCIÓN

Podríamos orientar esta unidad sobre autómatas simplemente hacia su uso, aprender una serie de operaciones, funciones y relaciones; cómo funcionan, cómo se programan y ya está.

Sin embargo no nos detendremos en saber si funciona, sino que daremos el salto a saber cómo funciona, porque no somos usuarios, somos técnicos.

En una máquina industrial, el usuario ve esto:

Figura 1. Mandos de control de una máquina industrial

Mientras que un técnico, lo que ve es esto:

Figura 2. Vista de la aparamenta de un armario eléctrico

De la misma manera, debemos conocer cómo funciona internamente un autómata. Empezaremos viendo cómo trata la información.

2. LA INFORMACIÓN EN EL AUTÓMATA

2.1. LA UNIDAD BÁSICA DE INFORMACIÓN. EL BIT

Diariamente, nos encontramos con situaciones que implican decisiones y estados, causas y consecuencias.

2.1.1. ¿QUÉ ES UN BIT?

Tanto en entornos industriales como en entornos domésticos estamos rodeados de máquinas que presentan un estado de activación.

A nuestro alrededor podemos encontrar máquinas en entornos industriales.

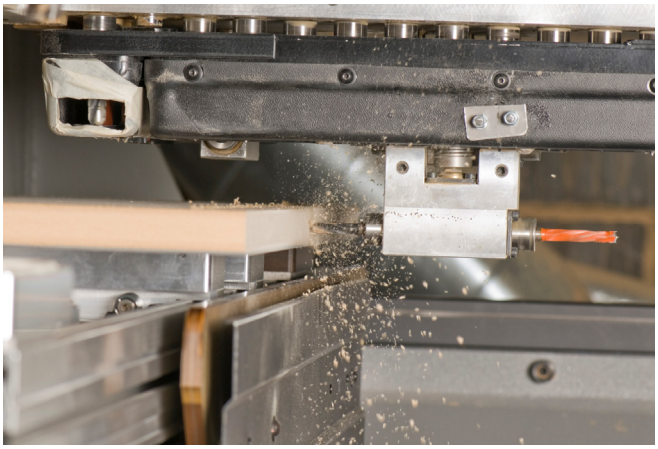


Figura 3. Máquina industrial

Y también nos encontramos con máquinas en nuestro entorno doméstico.

Figura 4. Lavadora doméstica

Estos dispositivos los podemos encontrar funcionando o parados.

Ante cualquiera de estas dos situaciones nos podemos plantear la siguiente pregunta:

¿Está activado?

La repuesta puede ser:

Sí está activado.

No está activado.

He aquí un dato, una información. A este dato, que es el estado de activación le corresponden dos posibles situaciones:

Sí: Verdadero

No: Falso

Si buscásemos un dato que solo nos ofreciese un estado de activación, este no podría cambiar. Perdería interés como información, pues sería totalmente predecible, como si en un periódico contasen todos los días las mismas noticias: no tendría utilidad.

Un dato que presenta dos estados es:

ACTIVO – NO ACTIVO

VERDADERO – FALSO

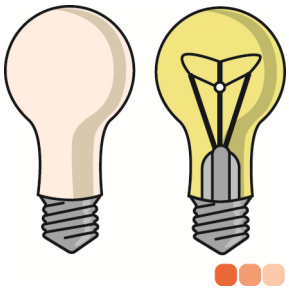
TODO – NADA

1 – 0

Esta es la unidad elemental de información. Su nombre, **bit**.

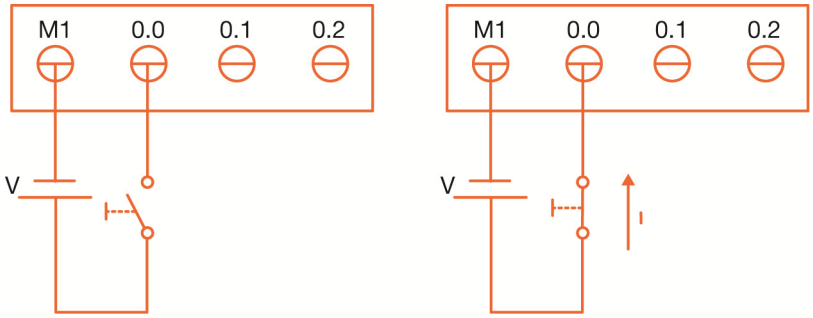
2.1.2. El bit y los sistemas electrónicos

Una unidad de información que nos ofrece dos estados posibles tiene la ventaja de poderse representar de una forma muy sencilla. Cualquier magnitud o cosa que pueda estar o no estar sirve para representar o expresar un bit.



Una bombilla apagada o encendida presenta dos posibles estados.

Un vaso vacío o lleno presenta dos posibles estados.



Un circuito eléctrico sin corriente (abierto) o con corriente (cerrado) presenta igualmente dos posibles estados.

Los sistemas electrónicos adoptan la corriente eléctrica como vehículo transmisor de la información binaria, el bit.

Hay corriente -------- 1

No hay corriente ---- 0

2.1.3. La información estado-acción

La importancia de obtener una información sobre un estado se fundamenta en la necesidad habitual de tomar decisiones.

Figura 5. Un mapa nos proporciona una información concreta

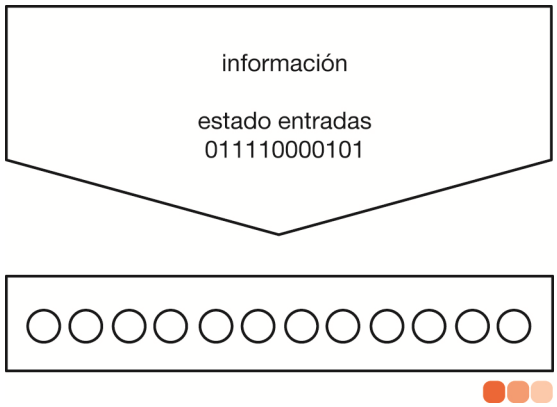
Al consultar un mapa estamos accediendo a una información de tipo geográfico. Esta información debe servir para acceder a un lugar. Es decir, en función de la información que nos proporciona el mapa nosotros realizamos la acción de decidir por dónde vamos.

La información es el primer paso para poder tomar decisiones.

Este es, a grandes rasgos, el principio de funcionamiento de un autómata. Para llevar a cabo el trabajo para el que ha sido programado, un autómata debe recibir en todo momento el estado o las acciones que realiza la instalación o máquina que está controlando, para llevar a cabo las acciones que hacen evolucionar el sistema hacia el comportamiento prefijado por el programa.

Veamos cómo es este proceso:

a) El autómata recibe una información del estado binario de las entradas:



El autómata recibe los estados de las entradas a través del cableado de las mismas.

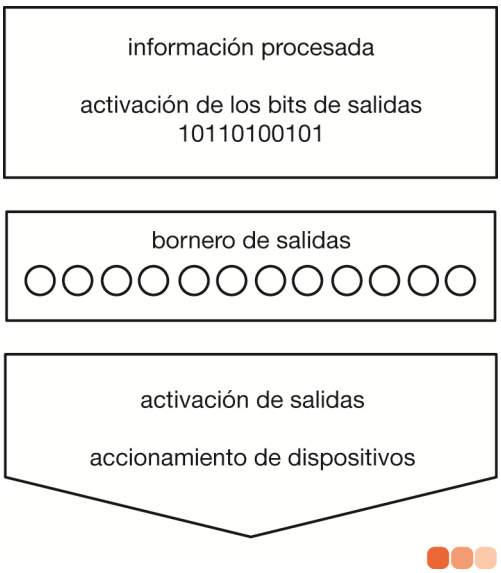
Hay corriente ---------- 1

No hay corriente ---------- 0

b) El autómata procesa la información de entradas, junto con otras informaciones internas como pueden ser temporizadores, contadores, etcétera. Ese procesamiento de la información recibida da lugar a la toma de decisiones, lo cual implica llevar a cabo acciones, que, en caso del autómata, supone la activación de salidas.

c) Activación de salidas: son las acciones que el autómata lleva a cabo en función de la información, tanto externa como interna, que en cada momento de la ejecución del programa está recibiendo. Hay que destacar que la ejecución del programa es permanente; en todo momento, se están tomando decisiones por parte del autómata. Estas decisiones se traducen en una serie de salidas de las que dispone el autómata y que permiten el accionamiento de los dispositivos de la instalación o máquina a controlar.

   Estas acciones se expresan, al igual que las salidas, en bits. Cada bit, una salida.



BIT	SALIDA
1	Hay corriente. Circuito eléctrico de salida cerrado
0	No hay corriente. Circuito eléctrico de salida abierto

2.2. AGRUPACIONES DE BITS

Como hemos visto, un bit es capaz de representar un estado de activación:

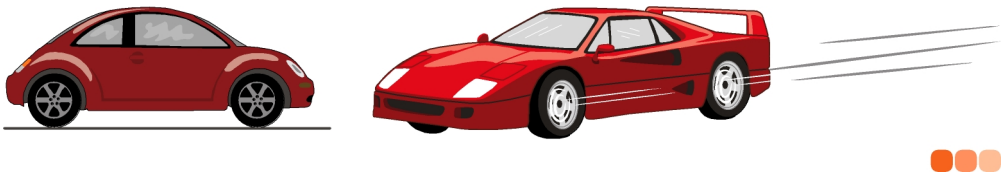


Figura 6. Un automóvil puede estar parado o en marcha

Podemos representar el estado del coche parado ---------- 0

Podemos representar el estado del coche en marcha -------- 1

Pero, ¿si nos preguntan qué velocidad lleva?

No sabríamos expresarlo con un bit. En realidad es imposible, porque nos están solicitando una información que necesita ser expresada de una forma mucho más compleja que con un simple bit, que solo nos puede informar sobre dos posibles estados.

Por supuesto, esto no quiere decir que todo lo visto sobre los bits sea muy limitado y carezca de utilidad. Ya que si bien un bit solamente nos ofrece dos posibles estados, no solo tenemos que trabajar con un bit, podemos trabajar con muchos. En realidad, para representar o expresar datos más complejos o variados se utilizan muchos bits, pero organizados de forma ordenada. Trabajaremos, entonces, con agrupaciones de bits, de forma que, combinándolos por medio de reglas, podamos expresar datos y valores, aparentemente vetados a la expresión binaria.

2.2.1. EL *BYTE*

Cualquiera que haya oído hablar alguna vez de ordenadores, con respecto a sus características, habrá escuchado referencias a la memoria en términos de “no sé cuántos *kas*” o “*megas*”. Estas cantidades son los múltiplos del *byte*.

1 *ka* es 1 kB = 1.024 B

1 *mega* es 1 MB = 1.024 kB = 1.048.576 B

Es decir, el *byte* se utiliza como unidad para medir la cantidad de memoria que tiene un dispositivo eléctrónico, como es el caso de los ordenadores, pero también el de un reproductor MP3 en el que escuchas tu música favorita o una cámara de fotos digital.

En el caso de los autómatas, si un autómata tiene, por ejemplo, una memoria de programa de 16 kB, eso quiere decir que podemos hacer programas que ocupen como máximo:

16 x 1.024 *bytes* = 16.384 *bytes*

Un *byte* es una agrupación de bits. Esto nos permitirá representar con más posibilidades que un simple 0 ó 1. Un *byte* está formado por 8 bits.

Esta agrupación de 8 bits nos permite codificar valores en código binario. Estos valores tienen su expresión o "traducción" en decimal.

7	6	5	4	3	2	1	0
1	1	1	0	0	0	1	0

Este sería un *byte* en el que hemos codificado en binario el valor decimal 226.

Comprueba la correspondencia entre las expresiones decimal y binaria del número 226.

Otra forma más simple de expresar nuestro *byte* sería la siguiente:

11100010

Presentando únicamente los 1 y 0.

También podríamos expresarlo en hexadecimal:

E2

Como hemos apuntado anteriormente, hay muchos casos en los que la memoria de un dispositivo "ocupa" muchos *bytes*. Entonces se acude a múltiplos del *byte*.

Un múltiplo del *byte* es el *kilobyte* (kB).

Cada kB corresponde a 1.024 *bytes.*

1 kB = 1.024 *bytes.*

Si la cantidad de memoria excede del millón de *bytes,* surge un nuevo múltiplo: el *megabyte* (MB).

1 MB = 1.024 kB = 1.048.576 *bytes.*

Mayores necesidades de memoria hacen que hablemos del *gigabyte* (GB).

1 GB = 1.024 MB = 1.048.576 KB = 1.073.741.824 *bytes*.

Todo esto está muy bien: vemos que los múltiplos de *bytes* permiten crear bancos de memoria donde almacenar muchos millones de *bytes*. Pero, ¿qué podemos expresar con un *byte*? ¿Cuántos valores decimales distintos caben en un *byte*?

Muy sencillo, veamos el menor valor posible:

0	0	0	0	0	0	0	0

El cual corresponde al valor decimal 0.

Veamos ahora el mayor valor posible:

1	1	1	1	1	1	1	1

El cual corresponde al valor 255; luego podemos expresar 256 posibles combinaciones, desde 0 hasta 255.

O dicho de otra manera:

Base binaria---------- **2**

Número de bits--------- **8**

Número de posibles combinaciones: $\mathbf{2^8 = 256.}$

2.2.2. *Word*

Como hemos comprobado, por medio de un *byte* solamente podemos expresar hasta el valor decimal 255.

Pero imaginemos que necesitamos expresar valores mayores: ¿cómo expresamos valores mayores de 255?

Para superar esta dificultad surge una nueva unidad, la **palabra** (o *word*, en inglés).

Word es un registro formado por 16 bits, es decir, 2 *bytes*.

La representación de un *word* será la siguiente:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	0	1	0	0	1	0	1	1	0	1	0

El número de orden de los bits dentro del *word* será desde el **0** hasta el **15**.

Podemos hacer la representación más simple:

1001101001011010

En hexadecimal será:

9A5A

Y ambos casos representan el valor decimal:

39.514

En cuanto al número de valores posibles expresables en un *word*, gracias a la combinación de sus 16 bits, razonaremos como para el caso ya visto del *byte* y sus 8 bits:

Byte $2^8 = 256$

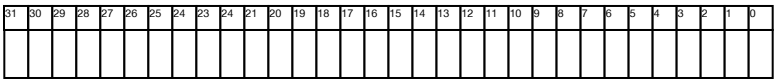
Word $\mathbf{2^{16} = 65.536}$

2.2.3. DOBLE *WORD*

Con el uso del *word* podemos expresar valores muy grandes, hasta 65.536. ¿Y si tenemos que expresar valores mayores? Para poder dar solución a este problema surge otra unidad superior: el doble *word*.

Doble *word* es un registro de 32 bits, es decir, 2 *words* o lo que es lo mismo, 4 *bytes*.

Al igual que los anteriores, un doble *word* es un registro binario que se representa teniendo en cuenta un número de orden o posición y un valor para esa posición.



El número de posibles valores representados será...

Un momento, vamos a dejar que seas tú quien lo averigüe:

Calcula el número de posibles valores representados por un doble *word* (32 bits).

Solución:
2^{32} = 4.294.967.296.

3. EL INTERIOR DEL AUTÓMATA

3.1. BLOQUES COMPONENTES DEL AUTÓMATA

El autómata está formado por tres bloques fundamentales y claramente diferenciados:

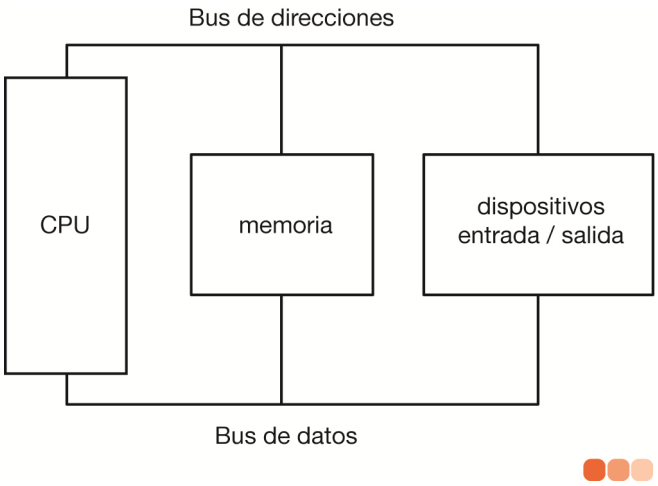


Figura 7. Bloques de componentes de un autómata

Estos elementos o bloques componentes del autómata son, como ya hemos visto en el gráfico:

- CPU: unidad central de proceso (*central processing unit*).
- Memoria.
- Dispositivos de entrada/salida.

Estos bloques, a su vez, están relacionados gracias a los buses. Los buses pueden ser de datos, de direcciones y de control.

3.2. DISPOSITIVOS DE ENTRADA/SALIDA

La función de los dispositivos de entrada/salida es proporcionar un medio de comunicación entre el exterior de la CPU o del propio autómata y dicha CPU.

Estos dispositivos pueden ser:

- Bornero de entradas del autómata.
- Bornero de salidas del autómata.
- Periferia descentralizada.
- Paneles de operador.
- Programadora o PC con *software* de programación para el autómata.
- Y en general, cualquier dispositivo que conecte el autómata con el sistema o máquina sobre la que ejerce el control.

Los dispositivos de entrada/salida son, de alguna manera, los "relaciones públicas" del autómata, los elementos que se relacionan con el exterior y toman información útil para la CPU.

El bloque de dispositivos de entrada/salida representa la gestión de la información entre la CPU y el exterior del autómata, así como el correcto direccionamiento de las decisiones tomadas por el autómata durante el procesamiento del programa.

La correcta gestión de la información es fundamental para un óptimo funcionamiento del programa. Con información errónea o mal direccionada, el resultado nunca será el esperado.

3.3. LA CPU

La función de la CPU es ejecutar y procesar la información guardada en la memoria. Para ello necesita un programa coherente y unos datos de información provenientes del exterior.

Para ello, la CPU se conecta a los dispositivos de entrada/salida (I/O) y a la memoria mediante un conjunto de cables denominados buses.

La CPU es el centro de cálculo del autómata. Es el integrante dentro del autómata que procesa los datos y toma las decisiones que afectan a la puesta en marcha del resto de los dispositivos y la instalación a controlar.

3.3.1. La CPU y la memoria

Para que la CPU pueda procesar correctamente la información, los datos deben estar guardados en la memoria del autómata. La información que necesita el autómata puede ser permanente o no permanente.

Ya veremos, cuando hablemos de la memoria, los tipos de memoria de que dispone el autómata. Por adelantar diremos que son básicamente dos tipos: permanente y volátil.

En la memoria permanente, la CPU busca información relacionada con la configuración del autómata. Esta memoria se denomina memoria ROM.

En la memoria volátil, la CPU busca información que puede cambiar con el tiempo, como puede ser el programa que nosotros le hayamos programado. La memoria volátil es conocida como memoria RAM.

3.3.2. Funcionamiento de la CPU

Un programa guardado en la memoria proporciona instrucciones a la CPU para llevar a cabo una acción. Dicha ejecución puede ser de distintos tipos, en función de la capacidad de proceso de dicha CPU: desde una simple operación aritmética, que podría hacer una calculadora, hasta el control de un proceso industrial.

La función de la CPU consiste en leer de la memoria el programa que le ha sido programado instrucción por instrucción, para ir ejecutando una a una las instrucciones que dicho programa le está indicando.

Por tanto, resulta vital que la CPU realice correctamente las acciones de lectura y ejecución de las instrucciones programadas. Para ello cuenta con una serie de recursos que veremos a continuación.

3.3.2.1. Recursos de la CPU

- **Número de registros a disposición de la CPU**. Este es un recurso de vital importancia, pues utiliza dichos registros para almacenar información de tipo temporal. Esta información puede ser datos a procesar o direcciones de datos en la memoria.

  En general, cuanta más información se pueda contener en estos registros, diremos que la CPU tendrá más capacidad, será mejor, puesto que podrá manejar más datos a la vez. De esta forma podrá ser más rápida en su ejecución.

  Los registros de la CPU pueden ser de 8 bits, 16 bits o incluso más.

- **Unidad aritmético–lógica**. En inglés, *arithmetic-logic unit* (ALU). Esta unidad de la CPU es la encargada de las cuentas.

  Este recurso de la CPU realiza las funciones de tipo aritmético. Cálculos y operaciones aritméticas (sumas, restas...). Así como también las operaciones lógicas como AND, OR...

- **Contador de programa**. Un programa es una sucesión de instrucciones dispuestas una detrás de otra. Durante el procesamiento del programa, la CPU lee cada instrucción, la ejecuta y después pasa a la siguiente instrucción. Esta tarea está tutelada por el contador de programa, que es el que va llevando a la CPU de instrucción en instrucción.

  La función del contador de programa es apuntar a la siguiente instrucción, que debe ejecutarse. Cada vez que se ejecuta una instrucción el contador de programa se incrementa para apuntar hacia la dirección de la siguiente instrucción que debe ser ejecutada.

- **Decodificador de instrucciones**. El decodificador de instrucciones se encarga de interpretar la instrucción en curso recogida por la CPU en la memoria. Esta instrucción se guarda de forma momentánea en un registro especial denominado registro de instrucciones.

  El decodificador es un especie de diccionario en el que se guarda el significado de cada instrucción, de forma que le cuenta a la CPU los pasos a seguir para ejecutar correctamente cada una de la instrucciones que tiene almacenadas. El conjunto de instrucciones que tiene la CPU se denomina **juego de instrucciones**.

  La CPU y por tanto el autómata, será mejor cuantas más instrucciones contenga su juego de instrucciones. Es decir, cuanto mayor sea el número de instrucciones que puede comprender.

- **Reloj**. El reloj es el encargado de indicar a la CPU cuando se pasa de una instrucción a la siguiente. Es en realidad el que marca el ritmo del programa.

Cuanto más rápido sea el reloj, es decir, la frecuencia de oscilación del reloj, más rápido se ejecutará el programa. Por tanto esta es una de las características de la CPU que nos dará también medida de la calidad de la CPU. A mayor frecuencia de oscilación del reloj, más rapidez de la CPU; por lo tanto, más capacidad para procesar más instrucciones en menos tiempo.

La frecuencia de oscilación del reloj de cada CPU es fija y siempre se encuentra dentro de los límites soportados por el resto de elementos de la CPU.

Un caso común es cuando oímos hablar de ordenadores en términos de, por ejemplo, Pentium 2200. Esto se refiere al reloj de su CPU y quiere decir que su frecuencia de oscilación es 2.200 MHz. Un Pentium 3,2 GHz (que es lo mismo que 3.200 MHz) será, lógicamente, bastante más rápido.

3.3.2.2. El ciclo de la CPU

El ciclo de la CPU S7-1200 ejecuta de forma cíclica una serie de tareas, en las que se incluye el programa de usuario, es decir, el programa que introducimos al autómata para que este lo procese. Dicha ejecución la denominamos ciclo, pues se repite continuamente.

Durante el ciclo, la CPU ejecuta las tareas siguientes:

1. Lectura de entradas.
2. Ejecución del programa de usuario.
3. Procesamiento de peticiones de comunicación de dispositivos externos a la CPU.
4. Autodiagnóstico de elementos del autómata. Comprueba en cada ciclo que todo está en perfecto estado de funcionamiento.
5. Escribir el resultado del proceso en las salidas.

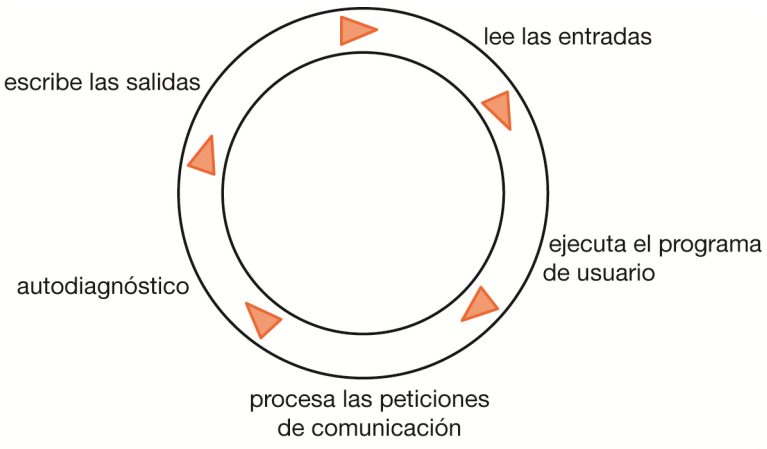


Figura 8. Ciclo de la CPU

La serie de tareas que se ejecutan durante el ciclo depende del modo de operación de la CPU. La CPU S7-1200 tiene dos modos de operación: *STOP* y *RUN*. Con respecto al ciclo, la principal diferencia entre *STOP* y *RUN* es que el programa se ejecuta al estar la CPU en modo *RUN*, pero no se ejecuta cuando está en *STOP*.

Recuerda que aunque en ocasiones te resulte dura esta fase del estudio, tu confianza, tu actitud positiva y tu perseverancia son los argumentos clave para el triunfo.

3.4. LA MEMORIA

Definición: la memoria es un dispositivo capaz de almacenar información.

La memoria sirve para almacenar información. Esta información puede ser nuestro programa o datos que este necesita. También puede ser información interna del autómata, que este necesita para su correcto funcionamiento, y a la que nosotros no tenemos acceso.

3.4.1. EL *BYTE* COMO UNIDAD FUNDAMENTAL

Ya conocemos el *byte*. Sabemos que es una unidad de memoria que está formada por 8 bits.

0	1	1	0	0	1	0	0

Vamos a suponer que esto es una “caja” en la que el autómata guarda información.

Pero, ¿cuántas informaciones distintas podemos guardar en esta caja?

Como ya vimos en apartados anteriores, el número de posibles combinaciones es **256.**

Comprueba la afirmación anterior sobre el número de posibles combinaciones en un *byte*.

Solución:
$2^8 = 256$.

En un *byte* podremos contener un solo dato cada vez; tendremos 256 posibilidades, pero solo un dato al mismo tiempo.

Para solucionarlo vamos a utilizar más *bytes*. Será como tener muchas cajas donde guardar información.

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**		
									***Byte* 0**
									***Byte* 1**
									***Byte* 2**
									***Byte* 3**
									***Byte* 4**
									***Byte* 5**
									***Byte* 6**
									***Byte* n-1**
									***Byte* n**

Añadiendo más *bytes* conseguimos una matriz de bits ordenados en **n** *bytes*.

Los autómatas programables y otros dispositivos eléctronicos programables, como los ordenadores, tienen una matriz de *byte* de este tipo. Esto es un banco de memoria.

Cada una de las casillas estará ocupada por un bit, cuyo estado podrá ser 0 ó 1.

Físicamente, en el caso de los dispositivos electrónicos el elemento utilizado para manifestar el estado del bit es un pequeño interruptor electrónico, el cual puede estar abierto (0) o cerrado (1). Estos interruptores son elementos semiconductores que están insertados en una pastilla de silicio.

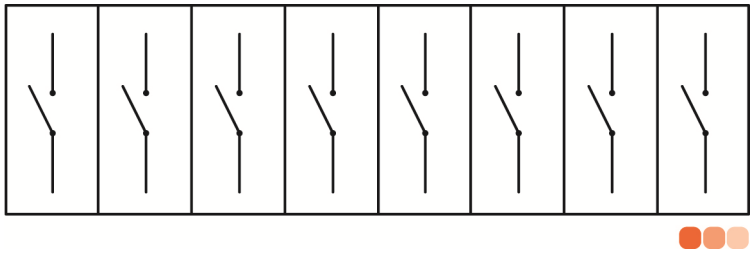
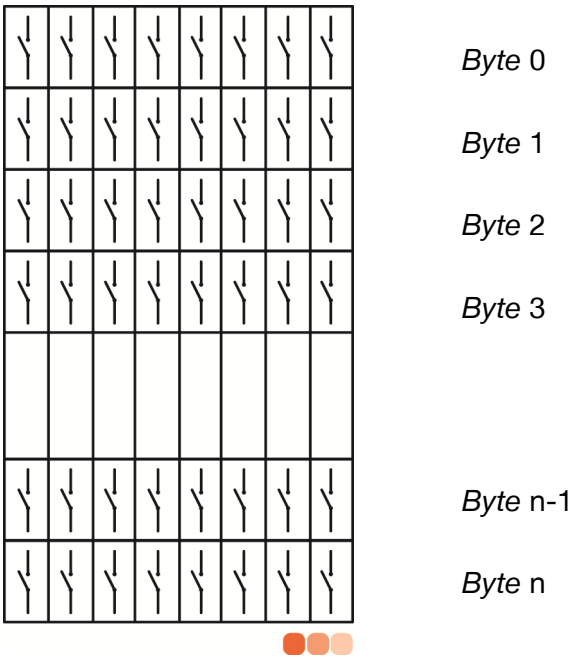


Figura 9. *Byte* formado por ocho microinterruptores

Si unimos **n** *bytes* de microinterruptores, tendremos una matriz de interruptores cuyo estado abierto-cerrado, generará una forma de memorizar datos. Los microinterruptores memorizan el valor de cada bit en su *byte* correspondiente.

La combinación de esos bits supone un valor almacenado en el *byte*. Y si tenemos **n** *bytes*, tendremos **n** datos almacenados en la memoria.



Matriz de microinterruptores. Su estado permite memorizar datos

De esta forma, al darle nombre o numeración a cada *byte*, y conociendo el bit correspondiente de ese *byte* (0-7), tenemos localizada toda la información que podemos almacenar en la memoria, tanto para modificarla como para leer en ella.

3.4.2. ALMACENAMIENTO DEL PROGRAMA EN LA MEMORIA

Una instrucción está formada por dos partes claramente diferenciadas: **operación** y **operando,** de forma que cada una tiene una función bien determinada:

- **Operación**: indica a la CPU "lo que debe hacer".
- **Operando**: indica a la CPU "sobre qué debe hacerse".

Una instrucción consta de dos zonas de información para el procesador o CPU:

Código de operación.

Dirección del operando.

CO	DO

DATOS BINARIOS. *BYTE*

Veamos ahora cómo almacena el autómata el programa en la memoria.

En la memoria del autómata solamente podemos almacenar datos en formato de bit, unirlos para formar *bytes* y tener datos más complejos. Y en todo caso, utilizar unidades mayores, pero siempre basadas en el *byte* y por tanto, en el bit.

Visto que dependemos en exclusiva del bit, trabajemos con él. Vamos a suponer que nuestro autómata es capaz de realizar ocho operaciones diferentes. Necesitaremos por tanto ocho combinaciones diferentes para hacer referencia a ellas.

Para obtener 8 combinaciones diferentes basta con utilizar 3 bits.

Vamos a establecer la operación a la que corresponderá cada combinación de esos 8 bits:

Operación	Operando		
FIN	0	0	0
SÍ ESTÁ ACTIVADO	0	0	1
O ESTÁ ACTIVADO	0	1	0
Y ESTÁ ACTIVADO	0	1	1
SI NO ESTÁ ACTIVADO	1	0	0
O NO ESTÁ ACTIVADO	1	0	1
Y NO ESTÁ ACTIVADO	1	1	0
ACTIVA A	1	1	1
	BIT 2	BIT 1	BIT 0

En esta tabla observamos cómo asignamos código de operación a cada una de las operaciones de la lista de instrucciones descrita. Vemos cómo codificamos mediante 3 bits cada operación, para así obtener 8 posibles operaciones distintas.

Si codificamos las operaciones con 1 y 0, ya podemos guardarlas en la memoria del autómata.

En lugar de 3 bits vamos a utilizar *bytes*, es decir, grupos de 8 bits. Aún nos quedan 5 bits para codificar el operando de cada instrucción.

Estos 5 bits nos ofrecen 32 combinaciones posibles para codificar direcciones de operando. Utilizaremos 16 para direccionar 16 entradas y las restantes para direccionar 16 salidas.

Por tanto, cada instrucción de nuestro programa constará de 1 *byte*. En el cual 5 bits se dedicarán a direccionar el operando de la instrucción y los 3 bits restantes se dedicarán a direccionar la operación que afecta a dicho operando.

CO	DO
3 bits	5 bits

CO: código de operación. Formado por 3 bits. Indica a la CPU qué debe hacer.

DO: dirección de operando. Formado por 5 bits. Dice a la CPU con qué debe hacerlo.

Veamos un pequeño ejemplo:

LISTA DE INSTRUCCIONES		CÓDIGO MÁQUINA							
Operación	Operando	CO			DO				
SÍ ESTÁ ACTIVADA	I0.0	0	0	1	0	0	0	0	0
Y ESTÁ ACTIVADA	I 0.1	0	1	1	0	0	0	0	1
ACTIVAR A	Q0.0	1	1	1	1	0	0	0	0
FIN		0	0	0	0	0	0	0	0

La parte indicada como código máquina será los bits que quedarán almacenados en la memoria del autómata. De esta forma tan sencilla, quedan guardadas las instrucciones en la memoria para que la CPU pueda leerlas y llevar a cabo la ejecución del programa.

Hasta ahora hemos hablado de memoria de forma muy genérica. Eso está bien como concepto inicial, para no abrumar con excesivos términos y poder consolidar mejor ciertos términos básicos o fundamentales.

Pero ha llegado la hora de profundizar un poco más en la memoria del autómata y conocer los distintos tipos de la misma contenidos en nuestro autómata.

3.4.3. ZONAS DE MEMORIA DEL AUTÓMATA

En la memoria del autómata se almacenan datos de muy diversa índole. Tanto en lo que se refiere a su origen como en lo referido a su destino. Estos datos pueden ser datos de configuración del autómata, el programa de usuario…

La memoria del autómata, a la que hasta ahora nos hemos referido de forma muy genérica, está acotada en diversas zonas según la función y el sentido de los datos almacenados en ellas.

Veamos un ejemplo de un mapa de memoria de un autómata en el que hay reservados un total de 2.048 *bytes* para almacenar datos.

Imagen del proceso de entradas I0 e I1 *2 bytes*	*byte* 0 --- *byte* 1
Imagen del proceso de salidas Q0 y Q1 *2 bytes*	*byte* 2 --- *byte* 3
Datos y programas internos del autómata (sistema operativo) *1.020 bytes*	*byte 4 --- byte1.023*
Memoria de programa Memoria reservada para almacenar las instrucciones que nosotros programamos *1.024 bytes*	*byte 1.024- byte 2.047*

3.4.3.1. Imagen del proceso de entradas-salidas

Los estados de señal de entradas y salidas quedan guardados como unos y ceros en las memorias de imagen del proceso de entradas y salidas del autómata.

Si nuestro autómata cuenta con 16 entradas y 16 salidas, cada una de estas zonas tendrá 16 bits, es decir, 2 *bytes* reservados para plasmar la imagen del estado de las entradas y salidas en dicho autómata. Así pues, cada entrada y cada salida tiene 1 bit asociado en la memoria del autómata.

Las entradas del autómata tendrán su bit asociado en la imagen del proceso de entradas.

Las salidas del autómata tendrán su bit asociado en la imagen del proceso de salidas.

Cuando al terminal físico de una entrada llega corriente eléctrica, el bit asociado a dicha entrada se pone a 1. Si no llega corriente, permanecerá a 0.

I0	I0.7	I0.6	I0.5	I0.4	I0.3	I0.2	I0.1	I0.0	*Byte* 0 de la memoria
I1	I1.7	I1.6	I1.5	I1.4	I1.3	I1.2	I1.1	I1.0	*Byte* 1 de la memoria

Supongamos que tenemos con tensión la entrada I0.0 y la entrada I0.6, así como la entrada I1.4. La memoria del autómata quedaría de la siguiente manera:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
I0	0	1	0	0	0	0	0	1	*Byte* 0 de la memoria
I1	0	0	0	1	0	0	0	0	*Byte* 1 de la memoria
	Imagen del proceso de salidas								
	Datos internos								

Para la imagen del proceso de salidas reservamos, al igual que en el caso de las entradas, 16 bits para expresar las 16 salidas de las que dispone el autómata. Por tanto 2 *bytes*.

Cuando el programa evoluciona de forma que el resultado de su ejecución es la activación de un bit de salida, el autómata hace que en el terminal correspondiente a ese bit de salida se cierre un circuito eléctrico que provoque la aparición de tensión en dicho terminal. Si el valor del bit está a 0, el terminal estará conectado a masa (sin tensión).

Q0	Q0.7	Q0.6	Q0.5	Q0.4	Q0.3	Q0.2	Q0.1	Q0.0	*Byte* 2 de la memoria
Q1	Q1.7	Q1.6	Q1.5	Q1.4	Q1.3	Q1.2	Q1.1	Q1.0	*Byte* 3 de la memoria

Supongamos que tenemos activadas las salidas Q0.2 y Q0.3, así como la salida Q1.6. La memoria del autómata quedaría de la siguiente manera:

	Imagen del proceso de entradas								
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Q0	0	0	0	0	1	1	0	0	*Byte* 0 de la memoria
Q1	0	1	0	0	0	0	0	0	*Byte* 1 de la memoria
	Datos internos								

3.4.4. Sistema operativo y datos del autómata

Esta zona de memoria supone la reserva que el autómata tiene para el sistema operativo del mismo. Es decir, para el conjunto de instrucciones internas que gobiernan el funcionamiento y las interrelaciones entre los distintos elementos que componen el autómata.

El sistema operativo incluye programas de sistema que fijan la ejecución del programa de usuario, la gestión de entradas y salidas, el reparto de la memoria, la gestión de datos y similares. El sistema operativo no se puede modificar.

A su vez, en esta memoria hay reservado un espacio para guardar datos sobre la configuración del autómata, valores intermedios durante la ejecución del programa, etc.

3.4.4.1. Memoria de programa

Tal y como indica su nombre esta memoria está dedicada a contener las instrucciones del programa que nosotros elaboramos. La CPU buscará en esta parte de la memoria el programa que debe ejecutar mientras permanezca en modo *RUN*.

En el ejemplo anterior hemos visto cómo el autómata contaba con 1.024 *bytes* de memoria de programa. Esto quiere decir que nuestro programa puede constar como máximo de 1.024 instrucciones. El tamaño de la memoria de programa es muy importante, puesto que limitará el tamaño de nuestros programas.

3.4.5. TIPOS DE MEMORIA

3.4.5.1. Memoria ROM (*read only memory*, memoria de solo lectura)

La memoria ROM es una memoria que permanece aunque desaparezca la fuente de alimentación a la que está conectada. Es una memoria remanente. Se conoce como memoria no volátil debido a esta característica. Hay distintos tipos de memoria ROM.

3.4.5.1.1. PROM (ROM programable)

Esta memoria es una memoria ROM que puede ser programada por el usuario. Cada bit de esta memoria se comporta de forma semejante a un fusible. La memoria se programa "fundiendo" estos fusibles. Esto quiere decir que una vez que hemos fundido ese fusible, no se puede volver a utilizar. En resumen, que es programable una sola vez. Por esto también es conocida como OTP (*one time programmable*, programable una sola vez).

3.4.5.1.2. EPROM (*erasable programmable ROM*, ROM borrable y programable)

La memoria EPROM sí que permite poder borrar y volver a grabar la memoria, es decir, permite la reutilización tantas veces como queramos. El problema que tiene es el sistema o método de borrado. Es necesario someterlas a luz ultravioleta para borrar la memoria y hacer que se puedan utilizar de nuevo. Esto, además de aparatoso, es lento, ya que puede llevar varios minutos borrar una pequeña memoria.

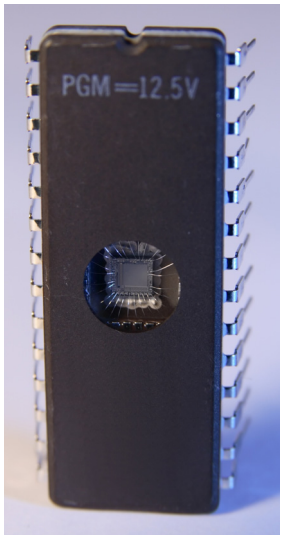


Figura 10. Memoria EPROM

3.4.5.1.3. EEPROM (*electrically erasable programmable ROM*, ROM programable borrable eléctricamente)

Como su propio nombre indica, esta memoria permite el borrado por medios eléctricos. A su vez el borrado es instantáneo, lo cual supone enormes ventajas frente a sus predecesores. Además, en esta memoria se puede seleccionar el *byte* que queremos borrar, algo que en las anteriores era imposible.

3.4.5.1.4. Memoria *flash* EPROM

Esta memoria debe su nombre a la rapidez con la que se realiza el proceso de borrado. Se borra en un *flash*.

El sistema de borrado es eléctrico, por lo que también se les conoce como *flash EEPROM*. La mayor diferencia con las EEPROM puras es que la *flash* se puede borrar completamente.

Figura 11. Un *pen drive* o las tarjetas de memoria de nuestra cámara digital son memorias *flash*

3.4.5.2. Memoria RAM (*random acces memory*, memoria de acceso aleatorio)

También es conocida como memoria volátil, ya que al quitarle la alimentación pierde su contenido. Hay tres tipos de memoria RAM.

3.4.5.2.1. SRAM (*static RAM* o RAM estática)

Cada una de las celdillas que forman parte de este tipo de memoria están formadas por biestables y no necesitan refrescar su contenido para mantener los datos.

3.4.5.2.2. DRAM (*dynamic RAM* o RAM dinámica)

Este tipo de memoria se caracteriza por el uso de un condensador para almacenar cada bit. El uso de condensadores tiene la ventaja de poder almacenar más información en el mismo espacio que las SRAM, pero tiene el inconveniente de que es necesario refrescar su contenido continuamente, debido a la pérdida de carga de los condensadores. Otra desventaja es que presentan un tiempo de acceso mayor a la información, es decir, son más lentas. Tanto en lectura como en escritura.

3.4.5.2.3. NVRAM (*non volatile RAM*, RAM no volátil)

Se trata de una memoria RAM a la que se le incorpora una pequeña batería integrada en el mismo chip. Al desconectar la alimentación eléctrica, la batería se encarga de mantener alimentada la memoria. La batería dura unos 10 años.

Date un respiro, pero no pierdas mucho tiempo, porque ya sabes que cuando algo tan importante como tu futuro está en juego... hay que ir a por todas.

3.5. EL BUS

La CPU se conecta a la memoria y a los dispositivos de entrada/salida mediante un conjunto de cables denominado bus.

Definición: el bus es el elemento que transporta la información, de forma similar a lo que haría un autobús.

Los dispositivos programables en general y por tanto nuestro autómata, disponen de tres tipos diferentes de buses: el bus de datos, el bus de direcciones y el bus de control.

Partimos del hecho de que la CPU controla el funcionamiento del autómata. Pues bien, para que un dispositivo del autómata, ya sea una memoria o un dispositivo de entrada/salida, sea reconocido por la CPU, debe tener una dirección asignada. Dicha dirección será única, no puede haber dos dispositivos con una misma dirección.

De la misma manera, tampoco puede haber un dispositivo con dos o más direcciones diferentes. La dirección es única e intransferible.

La CPU pone la dirección en el bus de direcciones; mientras, un circuito decodificador se encarga de encontrar el dispositivo al que se dirige. Una vez hecho esto, la CPU queda como enlazada al dispositivo y utiliza el bus de datos para tomar o enviar información a dicho dispositivo.

Para distinguir si la CPU recibe datos del dispositivo o los envía está el bus de control, que es el medio por el cual la CPU indica al dispositivo qué acción de las dos se va a realizar.

3.5.1. EL BUS DE DATOS

El bus de datos condicionará las prestaciones de la CPU. A mayor bus de datos, mejor rendimiento de la CPU, pues podrá manejar (enviar/recibir) más información en menos tiempo.

El bus de datos está formado por cables en paralelo, simple y llanamente. Cada uno de esos cables es una pista de información; sería como un carril de una autopista que nos lleva la información entre los dispositivos y la CPU.

Cada cable transmitirá una información binaria:

Hay tensión ---------- 1

No hay tensión ---------- 0

Si por cada cable circula un bit en un instante determinado, podremos transmitir tantos bits como cables tengamos a la vez.

Si nuestro bus consta de ocho cables, este bus permitirá que la CPU envíe o reciba un *byte* a la vez.

Cuanto mayor sea el bus de datos, es decir, cuantas más líneas tenga, más información podrá tratar a un mismo tiempo; por tanto, la comunicación de datos será más rápida.

El bus de datos es bidireccional, lo cual significa que la información puede viajar tanto desde la CPU a los dispositivos como desde los dispositivos a la CPU.

3.5.2. El bus de direcciones

Tal y como ya hemos visto, el bus de direcciones se usa para identificar los dispositivos y la memoria conectados a la CPU. Cuantos más cables tenga, mayor número de dispositivos se podrá direccionar. El número de dispositivos que podrá direccionar será el número de combinaciones que podrán realizarse con los bits que puedan emitirse a la vez por medio del *bus* de datos.

Es decir, si tenemos 8 cables disponemos de 256 combinaciones posibles. Por tanto, serán otros tantos los dispositivos que podremos direccionar.

El bus de direcciones, al contrario que el de datos, es unidireccional. Es la CPU quien envía una dirección al dispositivo para que este se prepare a comunicar.

3.6. Funcionamiento autómata

Veremos ahora cómo funcionan todos los elementos que hemos estado estudiando.

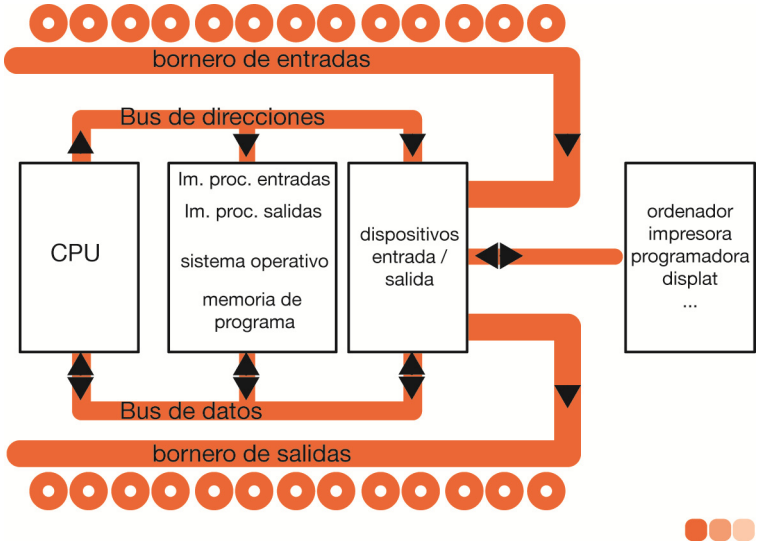


Figura 12. Flujo de la información entre la CPU y los dispositivos

Al conectar el autómata deberá estar todo preparado para que la CPU vaya a buscar a la zona de memoria del sistema operativo las instrucciones a seguir. Al arrancar el autómata, la CPU busca en el sistema operativo los programas para arrancar el funcionamiento.

El sistema operativo da a la CPU una serie de instrucciones:

1.º Le indica a la CPU dónde está la primera instrucción del programa de usuario. Es decir, le indica dónde está la primera posición de la memoria de programa.

2.º Le dice que tiene que leer el estado de las entradas mediante los dispositivos E/S, bornero de entradas. Las tensiones aplicadas o no en los terminales del bornero se convierten en 1 y 0 en la imagen de proceso de entradas.

3.º El sistema operativo indica a la CPU que tiene que ejecutar las instrucciones del programa de usuario, el que hemos programado. A su vez, la CPU va almacenando el estado de las salidas en la imagen del proceso de salidas.

4.º En este paso, la CPU actualiza las salidas. Convierte los 1 y 0 de la imagen del proceso de salidas en tensiones aplicadas en los terminales del bornero de salidas.

Y vuelta a empezar en un bucle cerrado que se repite mientras el autómata está en modo *RUN* (ejecución de programa).

CONCLUSIONES

En esta unidad hemos tratado los distintos tipos de memorias que tiene el autómata, cómo se relacionan entre ellas y cómo podemos utilizarlas.

Te proponemos que tomes la documentación técnica sobre el autómata S7-1200.

Busca en ella la información relativa a los componentes del autómata que hemos visto en esta unidad como tipo de CPU y memoria de cada una y exprésala en un listado.

RECAPITULACIÓN

- La unidad fundamental de información es el bit.
- El bit se agrupa en *byte*, *word* y *Dword* para expresar datos más complejos.
- En el autómata, la unidad de almacenamiento de memoria es el *byte* y sus múltiplos.
- La CPU gobierna el funcionamiento del autómata.
- Los dispositivos de entrada/salida relacionan la CPU con el entorno.
- La memoria sirve para almacenar la información en el autómata.
- En su funcionamiento, la CPU realiza un ciclo repetitivo de lectura, escritura y ejecución de programa.

AUTOCOMPROBACIÓN

1. El bit:

a) Presenta dos estados posibles.

b) Sirve para nombrar la CPU.

c) Son ocho *bytes*.

d) Solo puede ser una salida.

2. Un byte:

a) Está formado por 10 bits.

b) Está formado por 8 bits.

c) Es una entrada.

d) Es una salida.

3. El número de posibles valores representados por un *byte* es:

a) 250.

b) 128.

c) 256.

d) 8.

4. Un *word* está formado por:

a) 2 bytes.

b) 1.024 bytes.

c) 2 Dword.

d) 2 bits.

5. Un *Dword* está formado por:

a) 2 bytes.

b) 8 bits.

c) 4 words.

d) 4 bytes.

6. Un *word* está formado por:

a) 16 bits.

b) 20 bits.

c) 2 kB.

d) 4 *bytes*.

7. Un *megabyte* son:

a) 1.024 bytes.

b) 1.024 kB.

c) 10 words.

d) 1.024 bits.

8. La imagen del proceso de entradas:

a) Recoge en la memoria el estado de las entradas del autómata.

b) Indica la dirección de cada dispositivo de entrada.

c) Guarda el programa de usuario.

d) Nos indica el estado de las salidas.

9. La memoria EPROM:

a) No se puede borrar.

b) Se borra con luz ultravioleta.

c) Solo se graba una vez.

d) No se utiliza.

10. Las memorias de nuestra cámara de fotos:

a) Son memorias RAM.

b) Son memorias EEPROM.

c) No tienen memorias.

d) Son memorias *flash*.

SOLUCIONARIO

1.	a	**2.**	b	**3.**	c	**4.**	a	**5.**	d
6.	a	**7.**	b	**8.**	a	**9.**	b	**10.**	d

PROPUESTAS DE AMPLIACIÓN

Te proponemos que compruebes los tipos de memoria tipo *flash* que puedes encontrar en el mercado y los múltiplos de *byte* en los que se mide dicha memoria.

BIBLIOGRAFÍA

- MANDADO PÉREZ, E y otros. *Autómatas programables. Entorno y aplicaciones*. Madrid Thomson-Paraninfo, 2004.
- GONZÁLEZ RUEDA, E. *Programación de autómatas Simatic S7-300.* Barcelona: Ediciones Ceysa, 2006.